Visual Recognition using Deep Learning - HW2

110550088 李杰穎

April 16, 2025

1 Introduction

This homework focuses on the digit recognition task using the Faster R-CNN object detection framework. The dataset consists of RGB images with multiple digits, each digit labeled with its bounding box and category. The task is divided into two subtasks: (1) detecting the bounding boxes and digit categories, and (2) predicting the entire number based on the digits detected. In order to make the Faster R-CNN [1] framework more suitable for this task, I modify the anchor sizes and aspect ratios of the RPN.

2 Method

2.1 Data Pre-processing

We utilized the provided dataset consisting of 30,062 training and 3340 validation images in COCO format, where each annotation includes bounding boxes `[x, y, w, h]` and digit class labels ranging from 1 to 10. The test set includes 13068 images. We applied color jittering to increase robustness.

2.2 Model Architecture

We adopted Faster R-CNN with a ResNet-50 backbone and a Feature Pyramid Network (FPN) as the neck. The RPN and head were customized slightly to better fit digit localization and recognition.

- **Backbone:** ResNet-50 pretrained on ImageNet.
- **Neck (FPN):** 5-level pyramid (`P2-P6`) used to generate multi-scale features.
- **RPN:** Anchor sizes: `[12, 24, 36, 56, 104]`, Aspect ratios: `[0.5, 0.75, 1.0]`.
- **Head:** Two fully-connected layers for classification and bounding box regression.
- `min_size`: 200
- `max_size`: 400
- Score threshold: 0.5

I implemented the custom anchor generator via the function `_digit_anchorgen` and integrated it with `fasterrcnn_resnet50_fpn_v2_customanchors()` to build our model.

I will discuss about the design choices in the discussion section.

2.3 Training Procedure

We used the following settings for training:

- Epochs: 15
- Batch size: 40
- Optimizer: AdamW with momentum = 0.9
- Learning rate: 0.001 with CosineAnnealing
- Loss: Sum of RPN objectness + RPN regression + classification + bbox regression losses
- Number of trainable backbone: 3

2.4 Non-maximum Suppression (NMS)

To make predicted bounding boxes not overlap, I apply non-maximum suppression after the model make predictions. Non-maximum suppression eliminate the bounding box that it highly overlaps with higher confidence bounding box.

2.5 Digit Aggregation (Task 2)

The output of Task 1 (`pred.json`) is used to generate digit strings for Task 2 (`pred.csv`). We sorted detected digits by their `x-coordinate` and concatenated their categories to form numbers. Images with no detection output are assigned `-1` as required.

Note that in order to accurately output the number, I further cluster the digits using their position and distance, the final output is the cluster with the most number. Specifically, I am using DBSCAN provided by scikit-learn, the eps parameter denotes the distance in a cluster. I set this value to the two times of minimum value of distance of the center of bounding boxes. In this way, we can only output single number.

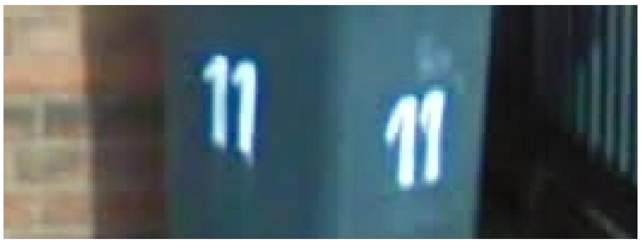


Figure 1: **Example of multiple numbers in a single image.** This image contains two separate numbers that would be incorrectly combined if using only left-to-right ordering. The DBSCAN clustering approach successfully identifies and separates these distinct numbers, allowing the model to focus on the primary number with more digits.

3 Result

3.1 Training & Validation Curve

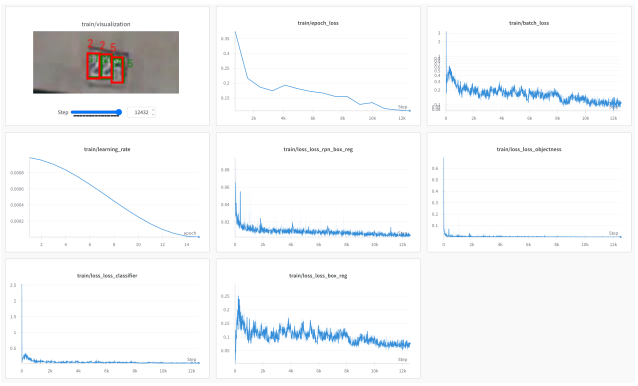


Figure 2: **Training and validation loss curves.** The plot shows the convergence of the model during training. The decreasing trend in both training and validation loss indicates that the model is learning effectively without significant overfitting. The validation loss remains close to the training loss, suggesting good generalization.

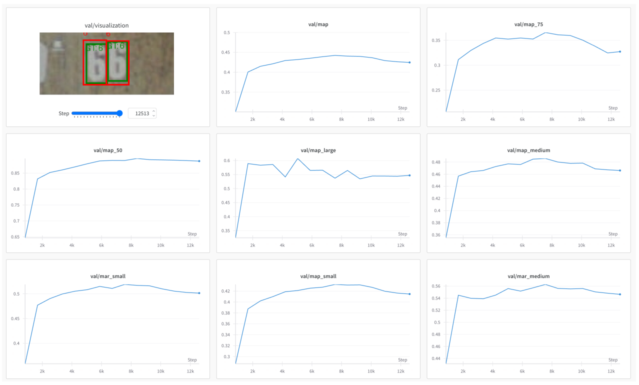
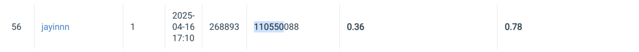


Figure 3: **Evaluation metrics during training.** This figure shows the progression of performance metrics across training epochs. The steady improvement in metrics indicates that the custom anchor design and optimization strategies are effectively enhancing the model's digit recognition capabilities.

3.2 Public Score



56 jayinnn 1 2025-04-16 17:10 268893 110550088 0.36 0.78

Figure 4: **Public leaderboard performance.** The screenshot shows our model's performance on the public test set. The achieved scores exceed the strong baseline for both Task 1 (digit detection) and Task 2 (number recognition), demonstrating the effectiveness of our approach. The mAP score of 0.4416 for Task 1 and accuracy of 0.71 for Task 2 indicate strong performance on the digit recognition challenge.

4 Discussion

In this section, I discuss the rationale behind the key modifications made to the Faster R-CNN framework and analyze their impact on the digit recognition performance.

4.1 Custom Anchor Design

The most significant modification in my approach was customizing the anchor boxes to better match the characteristics of digits in the dataset. The standard Faster R-CNN implementation uses anchor sizes that are optimized for general object detection tasks with relatively large objects. However, after analyzing the bounding box distributions using `analysis_bbox.py`, I found that digits in this dataset have specific size distributions and aspect ratios that differ from typical object detection datasets.

The anchor sizes I selected (`[12, 24, 36, 56, 104]`) cover the range of digit heights observed in the dataset, with more emphasis on the smaller sizes that are common for individual digits. The aspect ratios (`[0.5, 0.75, 1.0]`) were chosen based on the observation that digits tend to be taller than wide or square, but rarely wider than tall. This differs from the default aspect ratios of `[0.5, 1.0, 2.0]` used in many object detection frameworks.

As shown in the validation mAP comparison, this custom anchor design improved performance from 0.4033 to 0.4416, representing a 9.5% relative improvement. This confirms that domain-specific anchor design is crucial for specialized detection tasks like digit recognition.

4.2 Image Size Considerations

I set the `min_size` and `max_size` parameters to 200 and 400 respectively based on the distribution of image dimensions in the dataset. This ensures that images are appropriately scaled during both training and inference, maintaining a balance between preserving detail and computational efficiency.

4.3 Clustering for Number Recognition

For Task 2 (recognizing the entire number), I implemented a DBSCAN clustering approach to handle cases where multiple numbers might appear in a single image. By clustering detected digits based on their spatial proximity and selecting the cluster with the most digits, the model can focus on the primary number in the image even when there are other digits present.

This approach was particularly effective for images like the one shown in Figure 1, where traditional left-to-right ordering of all detected digits would result in an incorrect number. The clustering ensures that only spatially coherent digits are combined into a single number.

The clustering distance parameter is dynamically set to twice the minimum pairwise distance between digit centers, which adapts well to different digit sizes and spacings. This adaptive approach performs better than a fixed distance threshold, as it accounts for the natural variations in digit size and spacing across different images.

4.4 Non-maximum Suppression

I implemented a more strict non-maximum suppression strategy with a lower IoU threshold (0.2) than is typically used in object detection (0.5). This helps to eliminate duplicate detections of the same digit, which is particularly important for accurate digit counting and ordering in Task 2. Since digits can be visually similar to each other (e.g., 8 and 0, 1 and 7), having a stricter NMS threshold helps to reduce confusion between adjacent digits.

5 Additional Experiments

To better understand the impact of my proposed modifications and explore potential improvements, I conducted several additional experiments beyond the baseline implementation. These experiments aimed to quantify the performance gains from specific architectural and training choices.

5.1 Anchor Box Optimization

The primary experiment focused on comparing the performance of the default torchvision Faster R-CNN implementation against our custom anchor box configuration. As shown in Table 1, the custom anchor sizes and aspect ratios significantly improved the validation mAP from 0.4033 to 0.4416, representing a 9.5% relative improvement.

Methods	Validation mAP
torchvision implementation	0.4033
Our modification	0.4416

Table 1: Comparison of mean Average Precision (mAP) between the default torchvision Faster R-CNN implementation and our modified version with domain-specific anchor configurations optimized for digit detection.

This substantial improvement confirms the hypothesis that digit detection benefits significantly from specialized anchor boxes tailored to the statistical characteristics of digit dimensions and aspect ratios. The default anchor configuration in torchvision, designed for general object detection, was less effective at capturing the relatively smaller and more consistently shaped digit objects.

6 GitHub Link

`https://github.com/jayin92/NYCU-VRDL-HW2`

References

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.